Programming Tools

Advanced programming Python 3

Autor: Walter Rothlin

Stand: 27.9.2020

HWZ
Die Hochschule für Wirtschaft
in Zürich

Functions & Methoden

❑ Funktionen sind «selbständig»

❑ Funktionen «berechnen» anhand der Fct-Argumente (Parameter)einen Funktions-Wert (Return-Value) und haben keine Seiteneffekte (verändern von globalen Variablen)

❑ Zusammengefasst Funktionen in einem Module nennt man Libraries

❑ Call: `print('Hallo' + str(3.14) + 'BZU!', 3.14, sep='—', end='\n\n', flush=True)`

❑ Methoden gehören zu einer Klasse

❑ Methoden «verändern» oder «lesen» den Zustand (Instance-Variablen) eines Objektes

❑ Call: `'Dies {p1:12.2f} ist {place:20s}'.format(p1=3.1415, place='pi')`

Positional-Parameter & Optional-Parameter

```python
def sayHelloTo(firstname=None, lastname="Unknown", hellostr="Guten Morgen"):
    if firstname is not None:
        return "1:" + hellostr + " " + firstname + " " + lastname
    else:
        return "1:" + 'Hey you!'
```

```
Calls by Position
sayHelloTo()                                              1:Hey you!
sayHelloTo("")                                            1:Guten Morgen  Unknown
sayHelloTo("Walti")                                       1:Guten Morgen Walti Unknown
sayHelloTo("Walti","Rothlin")                             1:Guten Morgen Walti Rothlin
sayHelloTo("Walti","Rothlin", "Guten Abend")             1:Guten Abend Walti Rothlin
```

- ❑ Method-Overloading in Python
- ❑ Nach einem Optionalen Parameter nur noch Optionale Parameter
- ❑ Abwärtskompatibilität bei Erweiterungen

HWZ

Named-Parameter

```python
def sayHelloTo(firstname=None, lastname="Unknown", hellostr="Guten Morgen"):
    if firstname is not None:
        return "1:" + hellostr + " " + firstname + " " + lastname
    else:
        return "1:" + 'Hey you!'
```

Calls by Name
sayHelloTo(firstname="Max", lastname="Bi", hellostr="Gute Tag")   *1:Gute Nacht Max Bi*
sayHelloTo(hellostr="Hi", lastname="Roth", firstname="Walti")   *1:Hi Walti Roth*
sayHelloTo(firstname="XY")   *1:Guten Morgen XY Unknown*
sayHelloTo(lastname="R.", firstname="W.")   *1:Guten Morgen W. R.*

Calls by Position and Name
sayHelloTo("Walti",lastname="Rothlin", hellostr="Tag") *1:Tag Walti Rothlin*
sayHelloTo("Walti", "Rothlin", hellostr="Morgen,")   *1:Morgen Walti Rothlin*

❑ Zuerst Positional Parameter anschliessend nur Named-Parameter
❑ Abwärtskompatibilität bei Erweiterungen

HWZ

Operator-Overloading

```python
class Point:
    def __init__(self, xCoord=0, yCoord=0): # Ctr overloaded () (2) (2,3)
        self.__xCoord = xCoord
        self.__yCoord = yCoord

    # toString()
    def __str__(self):
        return "(" + str(self.__xCoord) + "/" + str(self.__yCoord) + ")"


    # operator overload +
    def __add__(self, point_ov):
        return Point(self.__xCoord + point_ov.__xCoord,
                     self.__yCoord + point_ov.__yCoord)

point1 = Point(2, 4)
point2 = Point(12, 8)
point3 = Point(2)
point4 = Point(yCoord=3)
print(point1, " + ", point2, " = ", point1 + point2) # (2/4) + (12/8) = (14/12)
```

Operator-Overloading (Math)

Operator	Expression	Internally
Addition	p1 + p2	p1.__add__(p2)
Subtraction	p1 - p2	p1.__sub__(p2)
Multiplication	p1 * p2	p1.__mul__(p2)
Power	p1 ** p2	p1.__pow__(p2)
Division	p1 / p2	p1.__truediv__(p2)
Floor Division	p1 // p2	p1.__floordiv__(p2)
Remainder (modulo)	p1 % p2	p1.__mod__(p2)

HWZ

Operator-Overloading (Logic)

Operator	Expression	Internally
less than	p1 < p2	p1.__lt__(p2)
Greater than	p1 > p2	p1.__gt__(p2)
Less than or equal	p1 <= p2	p1.__le__(p2)
Greater than or equal	p1 >= p2	p1.__ge__(p2)
Equal	p1 == p2	p1.__eq__(p2)
Not Equal	p1 != p2	p1.__ne__(p2)

HWZ

Operator-Overloading (Bitwise)

Operator	Expression	Internally
Bitwise Left Shift	p1 << p2	p1.__lshift__(p2)
Bitwise Right Shift	p1 >> p2	p1.__rshift__(p2)
Bitwise AND	p1 & p2	p1.__and__(p2)
Bitwise OR	p1 | p2	p1.__or__(p2)
Bitwise XOR	p1 ^ p2	p1.__xor__(p2)
Bitwise NOT	~p1	p1.__invert__()